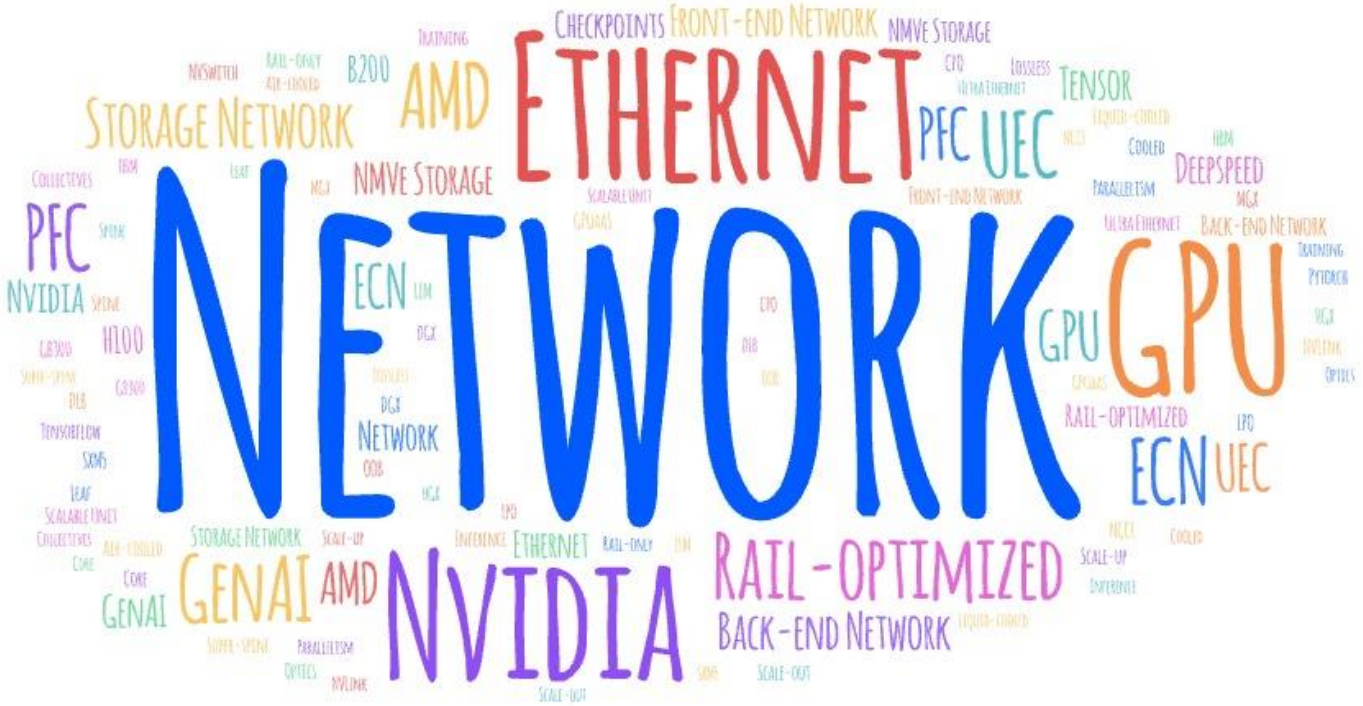
Building AI Clusters: What Network Engineers need to know

Brian Walsh, Head of IP PLM & Consulting Engineering, APAC

NOKIA

Networking for AI Clusters

Information Overload… What do I really need to know as a network engineer?



Building Distributed GPU Clusters

Key Building Blocks

Programing frameworks	PyTorch, TensorFlow, Horovod, deepspeed, RAY	
Cross-GPU operations	NCCL	RCCL
Networking within Nodes	NVLink, NVSwitch	Infinity Fabric
Networking across Nodes	Infiniband, RoCEv2, UEC	Infiniband, RoCEv2, UEC
GPU Kernels for AI Workloads	CUDA, CuDNN	ROCm
Accelerator	H100, B100/B200, GB200, B300, GB300	MI250, MI300, MI325X
	NVIDIA	AMD

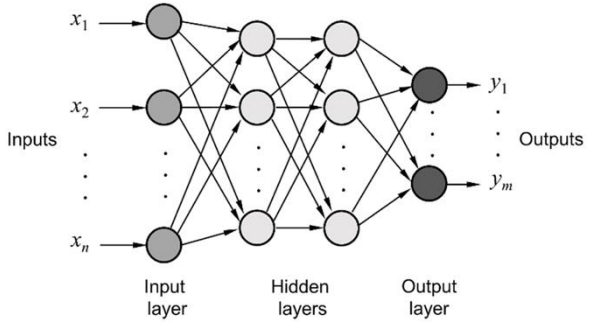
NOKIA

Building Distributed GPU Clusters

Programing frameworks

- Several open-source AI/ML programming frameworks are available to choose from. These provide developers with:
  - Simplified control over neural network architectures, loss functions and optimization algorithms with minimal code
  - User friendly programming interfaces that abstract the underlying hardware complexities
  - Integration with comprehensive ecosystem
    - Access to pre-built models, datasets, etc.
    - Monitoring & visualization tools for real-time tracking of training progress and resource utilization
  - Built-in support for distributed training
    - Scalability across multiple GPU's, nodes
    - Cross-GPU communication optimization
    - Data and model parallelization
  - Performance optimization techniques
    - Mixed precision training
    - Gradient accumulation

Multi-layer neural network

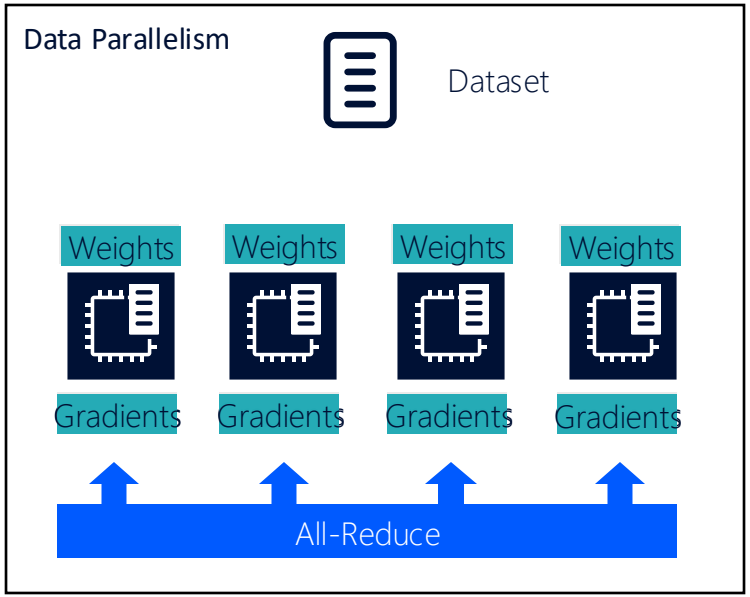


Pseudocode Example

```
import torch
import torch.distributed as dist
from torch.nn.parallel import DistributedDataParallel as DDP

# Initialize the process group with NCCL backend
dist.init_process_group(backend='nccl')

# Set the device for the current process
device = torch.device(f'cuda:{local_rank}')
torch.cuda.set_device(device)

# Create model and move it to the GPU
model = YourModel().to(device)

# Wrap the model with DDP (distributed device parallel)
model = DDP(model, device_ids=[local_rank])

# Training loop...
```

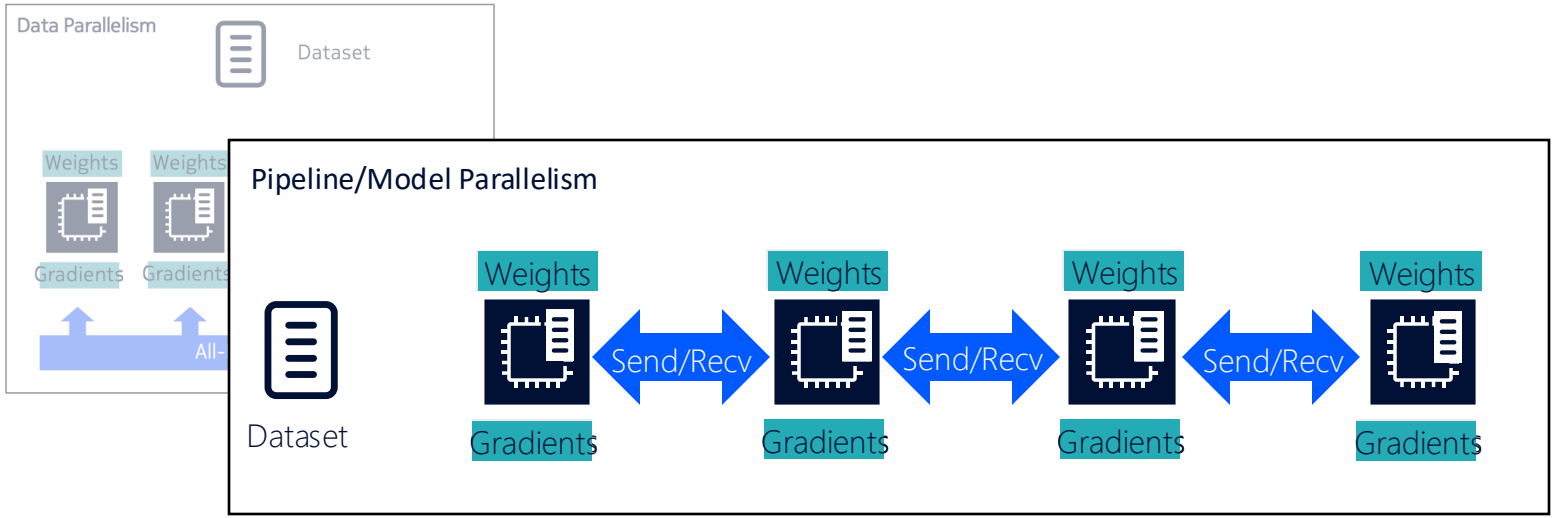
Building Distributed GPU Clusters

Distributed Training: GPU Collective Operations (e.g. NCCL, RCCL, etc.)



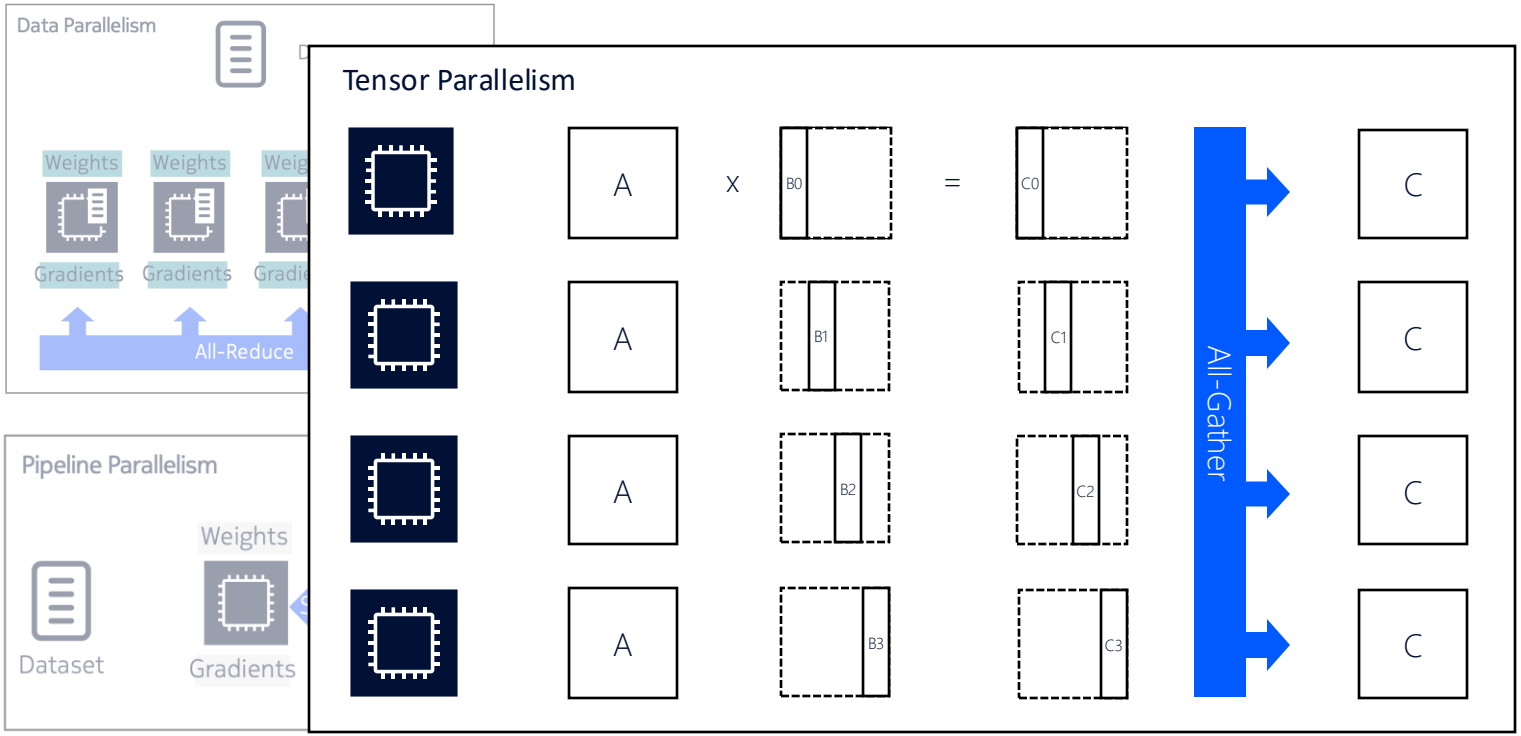
Building Distributed GPU Clusters

Distributed Training: GPU Collective Operations (e.g. NCCL, RCCL, etc.)



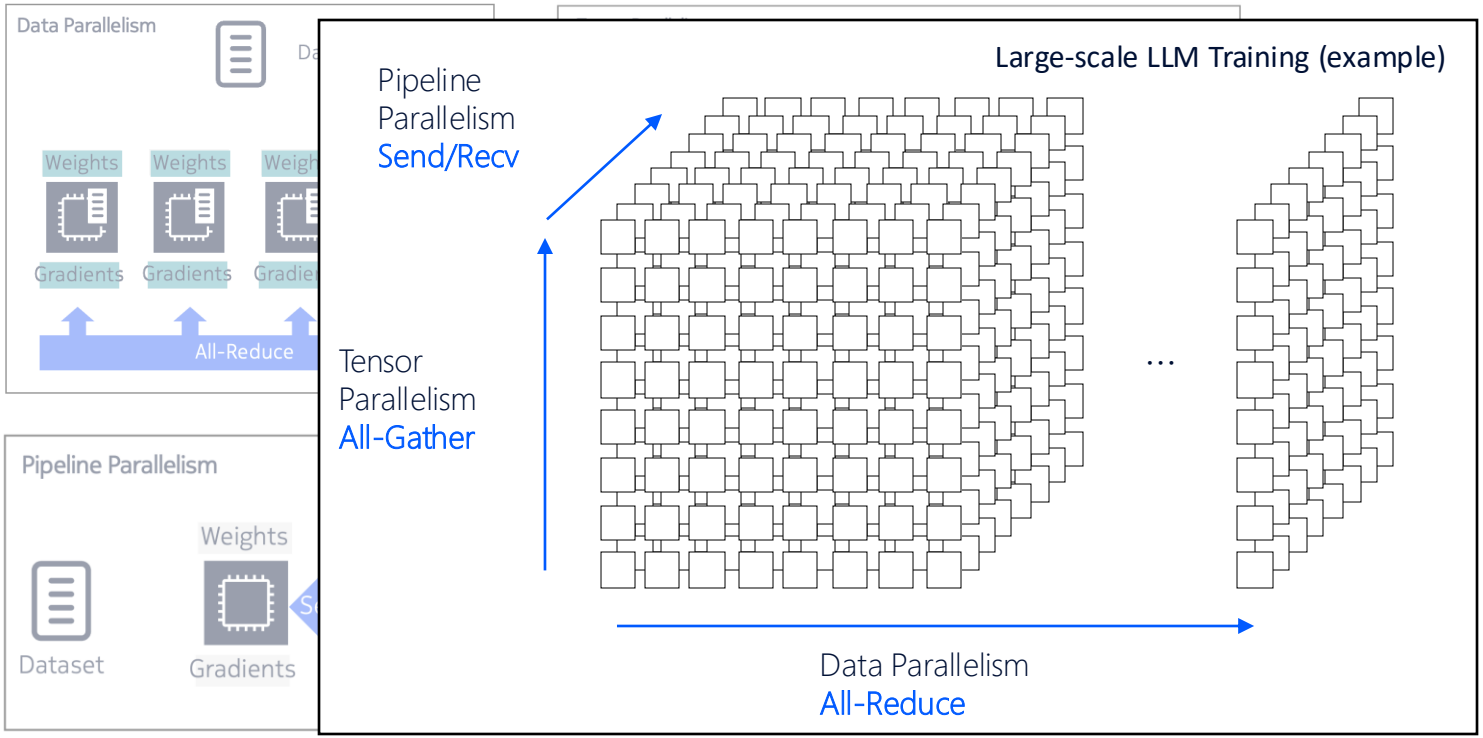
Building Distributed GPU Clusters

Distributed Training: GPU Collective Operations (e.g. NCCL, RCCL, etc.)



Building Distributed GPU Clusters

Distributed Training: GPU Collective Operations (e.g. NCCL, RCCL, etc.)



Building Distributed GPU Clusters

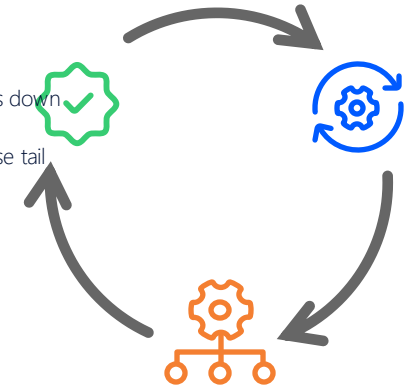
Distributed Training: GPU Collective Operations & Network Traffic Patterns

Synchronize

Parallel synchronization of parameters between GPUs

The slowest data arrival drags down the overall performance

JCT is based on the worst-case tail latency across the cluster



Compute

Distributed GPUs, which are interconnected with a network

Execute computationally intensive algorithms and perform calculations across huge datasets

Communicate

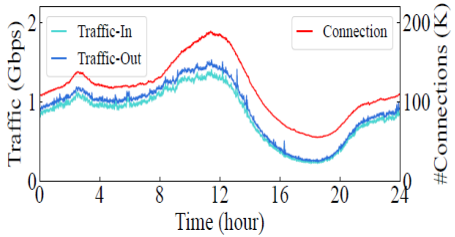
Every processor (GPU) exchanges parameters with every other processor

Data parallelism (DP)

Pipeline parallelism (PP)

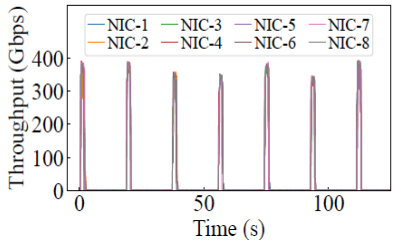
Tensor parallelism (TP)

Figure 1 – Traditional cloud computing traffic pattern



- Relatively continuous and steady, which slowly changes on the hourly scale

Figure 2 - NIC egress traffic pattern during production model training

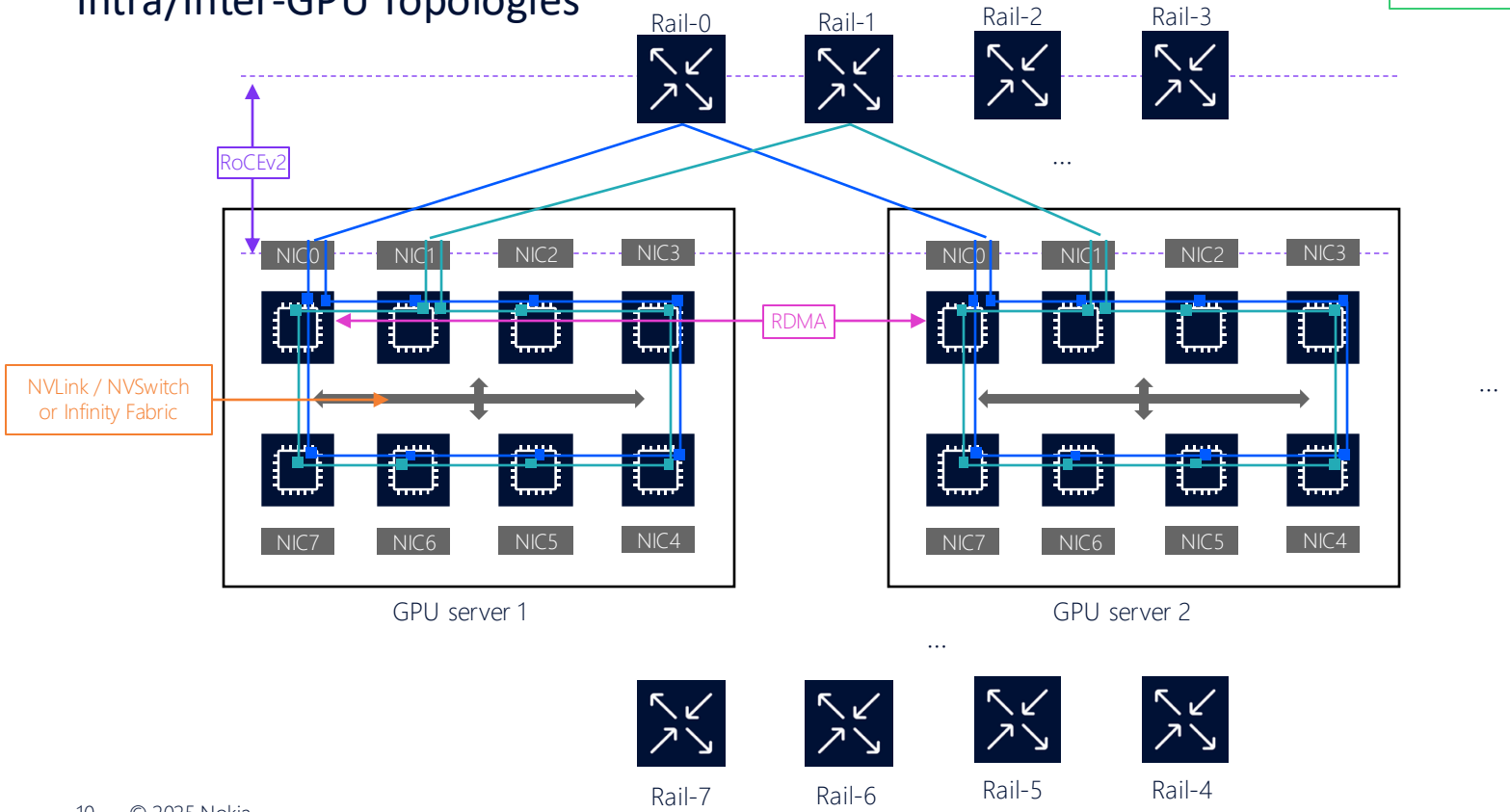


- Periodic burst in network utilization
- A few large flows (also known as having an elephant flow distribution)

Building Distributed GPU Clusters

Intra/Inter-GPU Topologies

Cross-GPU operations,
GPU topology calculations (Ring, Tree, etc.):
NCCL or RCCL



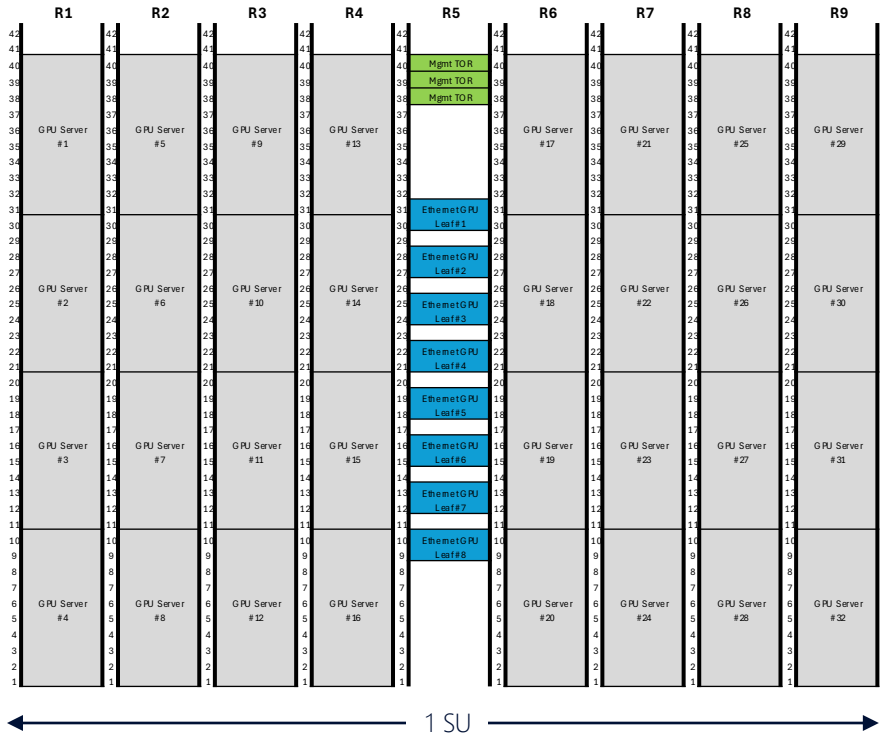
Building Modular AI Clusters

Scalable Unit (SU) (aka “stripe”)

- Operators typically build out clusters in modular units called SU’s (Scalable Units). This design simplifies deployment with the following key SU attributes
  - Self-contained mini cluster (e.g. 32 HGX nodes) pre-validated at line-rate across compute, storage and network
    - An SU’s GPU and storage sizing varies per customer, vendor, GPU generation, etc.
  - Repeatable power / thermal envelope, rack layout, etc. enabling predictable DC planning
  - Cabling, routing design, s/w versions are fixed per SU
  - Inside the SU, leaf switches interconnect GPU’s and storage via a single-hop
  - Optimal node count per-SU is closely linked to GPU server NIC count + Leaf SW radix
- GPU cluster scales out by deploying additional SU’s
  - Network architecture dictates how SU’s interconnect – typically via Spine/Core layer(s) (see following slides)

SU example:

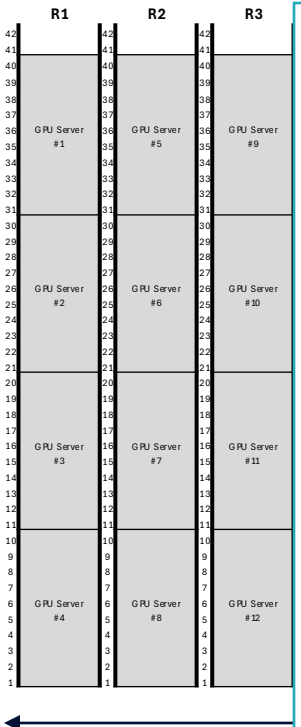
- 10 RU air cooled GPU servers
- 8 GPUs per server, 4 GPU servers per rack, 256 GPUs per SU
- 1 networking rack, 8 leaf switches, 3 mgmt. TOR switches
- 1 SU consumes 9 racks



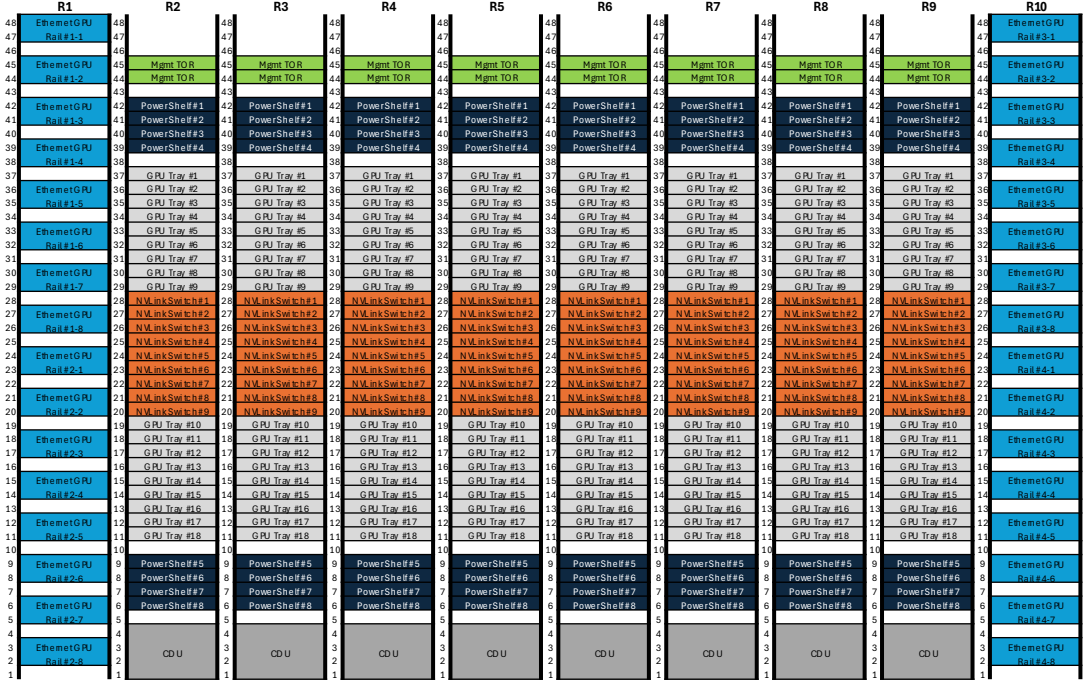
Building Modular AI Clusters

SU Configuration Flexibility

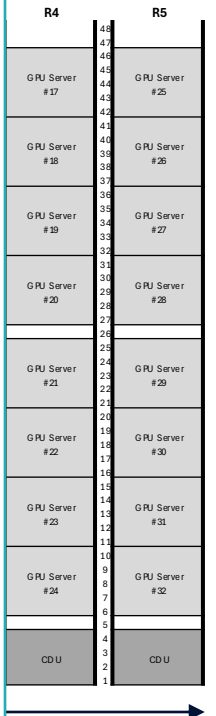
B200: 256 GPUs / air-cooled servers / 9 racks



NLV72/GB200: 576 GPUs / liquid-cooled servers / 10 racks

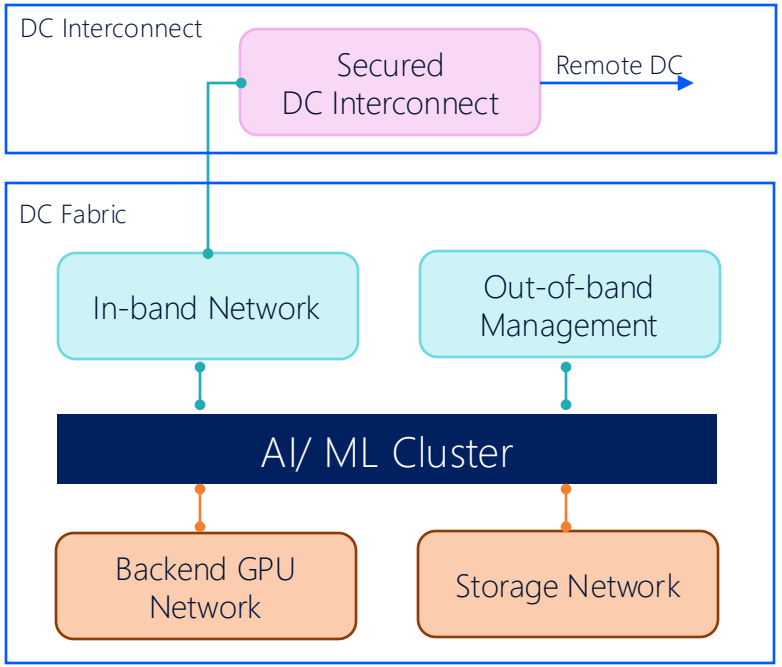


B200: 256 GPUs / liquid-cooled servers / 5 racks



AI Cluster Components

Different Types of Networks



GPU Backend & Storage Networks: Ethernet

- High-performance Ethernet profile required for specialized AI workloads
- Training and storage traffic is primarily RDMA
- Networks must support lossless Ethernet (RoCEv2) – providing flow control and optimal load balancing to proactively manage congestion
- GPU fabrics utilize specialized topologies designed to reduce latency
- No oversubscription
- Backend networks will begin to evolve to from RoCEv2 to UEC

In-band Network (aka Front-end): Ethernet

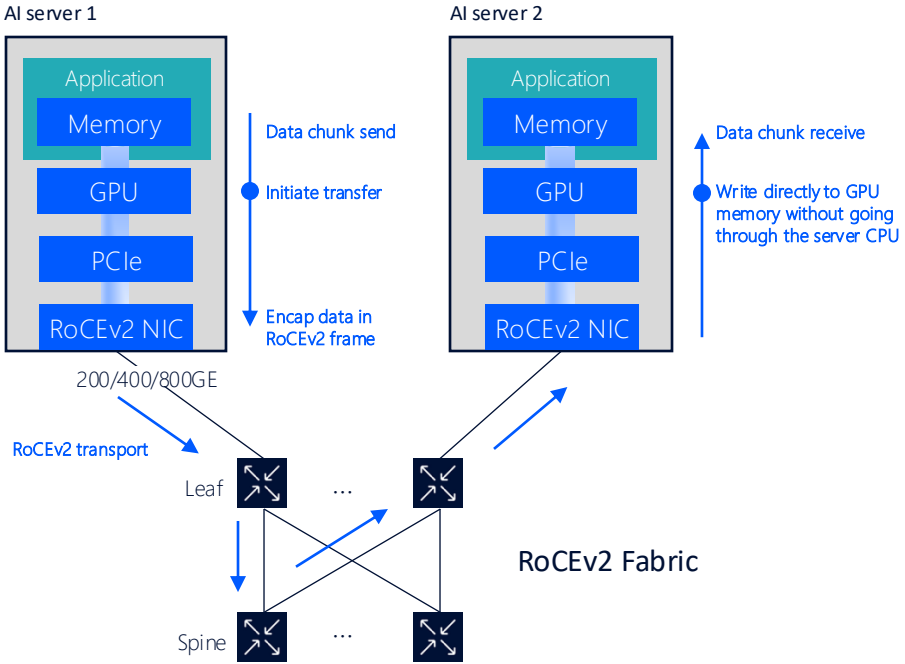
- Used for orchestrating training and for inference (i.e. user input/output)
- CLOS topology designed to connect x86 compute nodes to each other and to the outside world
- Oversubscription typical
- In-band and storage networks often combined in GPUaaS neoclouds

DC Interconnect Network

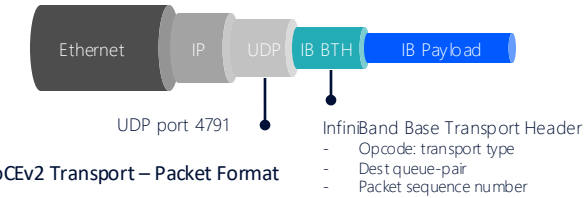
- WAN network to connect multiple Data Centers

Optimized GPU-to-GPU Data Transfers

RDMA over RoCEv2 Lossless Ethernet Fabric



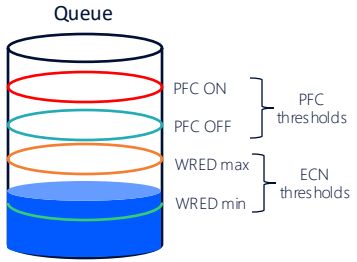
- RDMA (Remote Direct Memory Access) delivers the following performance attributes for massive data movements with DC environments:
  - Zero-copy data transfers
  - Kernel bypass
  - Low latency
  - High throughput
- RoCEv2 enables an InfiniBand transport layer to run over UDP/IP (IPv4 or IPv6) to support RDMA:
  - Flow are uniquely identified by Queue-Pair ID's
  - Relies upon a lossless Ethernet fabric

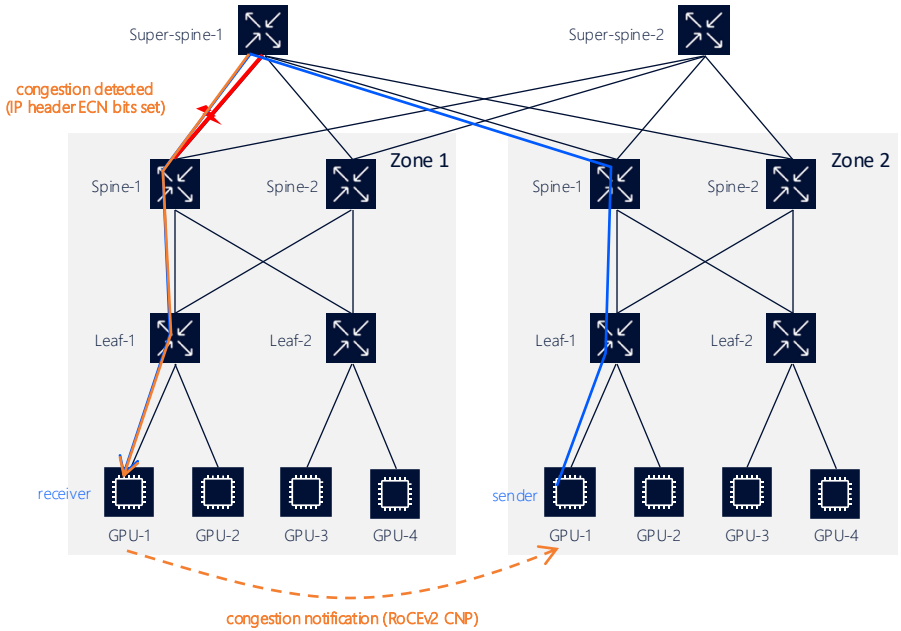


RoCEv2 Transport – Packet Format

RoCEv2 Lossless Ethernet Fabric

DCQCN (Data Cetner Quantized Congestion Notification) – ECN

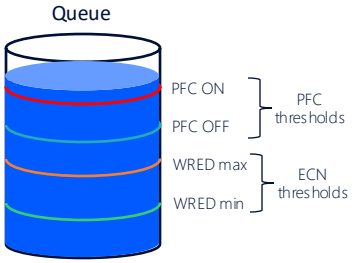


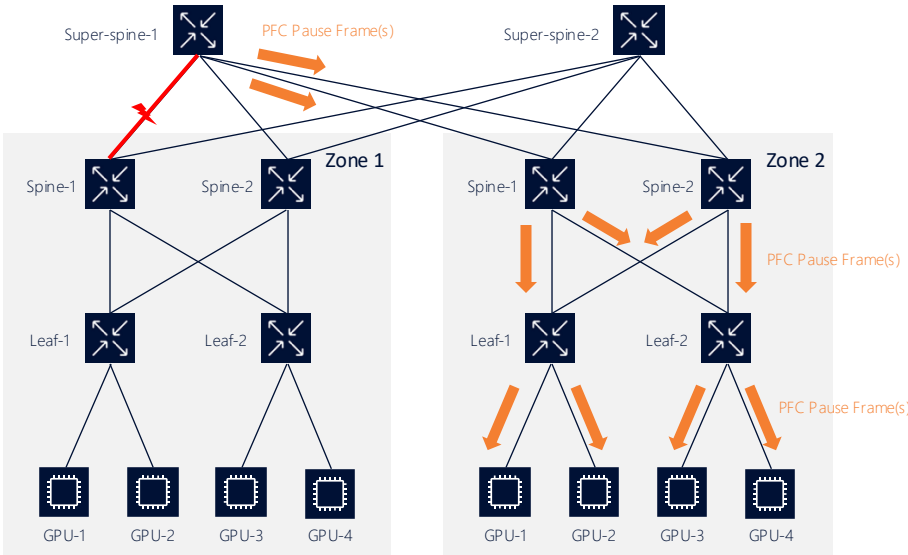


- ECN (Explicit Congestion Notification)
  - Provides end-to-end congestion control between two endpoints (RDMA NICs)
  - Upon interface congestion, DC fabric switches set the ECN bits (IPv4 ToS / IPv6 TC fields) of transiting packets to notify downstream receivers
  - ECN slope profile defines the proportion of packets marked w/ ECN bit
    - Q-depth < WRED min = no packets marked
    - WRED min < Q-depth < WRED max = linearly increasing proportion of packets marked
    - Q-depth > WRED max = all packets marked
  - During congestion, sender(s) reduce their transmission rate until congestion clears
  - Suitable to resolve minor congestion within DCF

RoCEv2 Lossless Ethernet Fabric

DCQCN (Data Cetner Quantized Congestion Notification) – PFC



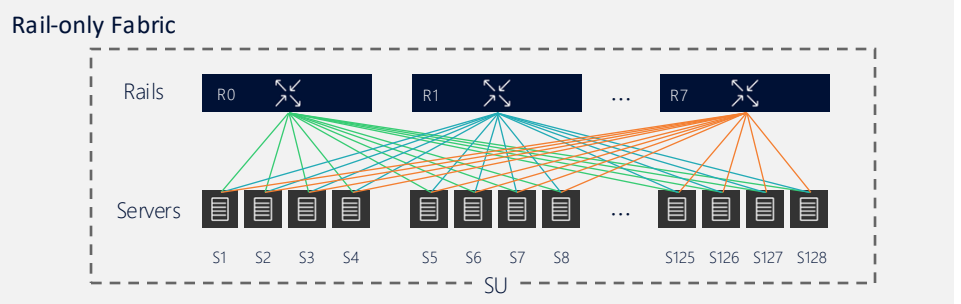


- PFC (Priority Flow Control) (IEEE 802.1Qbb)
  - Enables hop-by-hop per-priority flow control within the DC fabric and NIC's
  - When a buffer is becoming full, the switch/NIC requests the sender to temporarily stop sending traffic for a specific priority class (PFC pause frame)
  - The sender halts transmission until is receives a signal that the congestion has cleared
  - PFC ON/OFF thresholds define PFC pause frame behavior
    - Q-depth > PFC ON = send PFC pause frame to peer
    - Q-depth < PFC OFF = cease sending PFC pause frames
  - Suitable to resolve severe congestion (inc. microbursts) within DCF

Building Modular AI Clusters

GPU Network: Rail-Only Fabric

- Rail topology between servers and rail switches
  - Enables all GPU's within an SU domain to be reached within 1-hop
    - Rail-only topology enables large GPU domain w/ tightly bounded latency to improve JCT
  - GPU-0/NIC-0 to Rail-0, GPU-1/NIC-1 to Rail-1, etc. for all servers within each SU
  - Dependent on Nvidia PXN (PCI x NVLink) to utilize rails
    - Each node's local NVSwitch connectivity is used to first move traffic to the same rail as the destination GPU, then sending onto the destination so to not cross rails
    - Similar functionality exists for AMD clusters



- Design scale is bounded by rail switch radix
  - Examples:
    - 64p800GE → 128 GPU's w/ 400GE NIC (e.g. HGX B200)
    - 64p1.6T → 128 GPU's w/ 800GE NIC (e.g. HGX B300)
- Topology enables up to 1k GPU's assuming:
  - 8 x 64p800GE rail switches e.g. IXR-H5-64O
  - 8 x 400GE NIC's per server e.g. HGX B200
- If larger cluster scale is required, rail uplinks must be reserved and a rail-optimized topology deployed instead (see following slide)

* Ref: https://developer.nvidia.com/blog/doubling-all2all-performance-with-nvidia-collective-communication-library-2-12/

Building Modular AI Clusters

GPU Network: Rail-Optimized Fabric

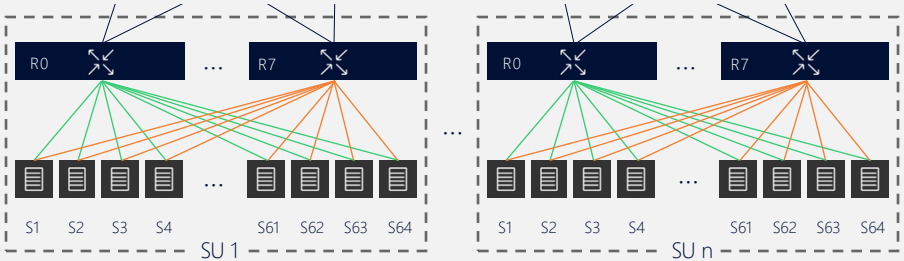
Intra-SU

- Rail topology between servers and rail switches
  - All GPU's within an SU domain can be reached within 1-hop
  - GPU-0/NIC-0 to Rail-0, GPU-1/NIC-1 to Rail-1, etc. for all servers within each SU
  - Optimized via GPU/server PXN capability to utilize rails instead of transiting spine layer
  - Single SU can support up to 512 GPU's assuming:
    - 8 x 64p800GE rail switches e.g. IXR-H5-64O
    - 8 x 400GE NIC's per server e.g. HGX B200

Intra-Zone

- Spine switches are organized into zones and connect rails belonging to subtended SU's
  - All GPU's within a zone can be reached within 3-hops
  - No oversubscription
  - Single zone can support up to 8k GPU's (same assumptions as above plus):
    - 64 x 64p800GE spine switches e.g. IXR-H5-64O
    - 128 x rail switches e.g. IXR-H5-64O

Rail-optimized Fabric



Inter-Zone

- Core switches are organized into planes and connect spines belonging to different zones
  - All GPU's within a cluster can be reached within 5-hops
  - No oversubscription
  - Spines within a zone connect with equal b/w into all planes
- Single cluster scales up to 262,144 GPU's
  - 2k x 64p800GE core switches e.g. IXR-H5-64O
  - 4k rail switches, 4k spine switches e.g. IXR-H5-64O

Legend

GPU-7/NIC-7

GPU-0/NIC-0

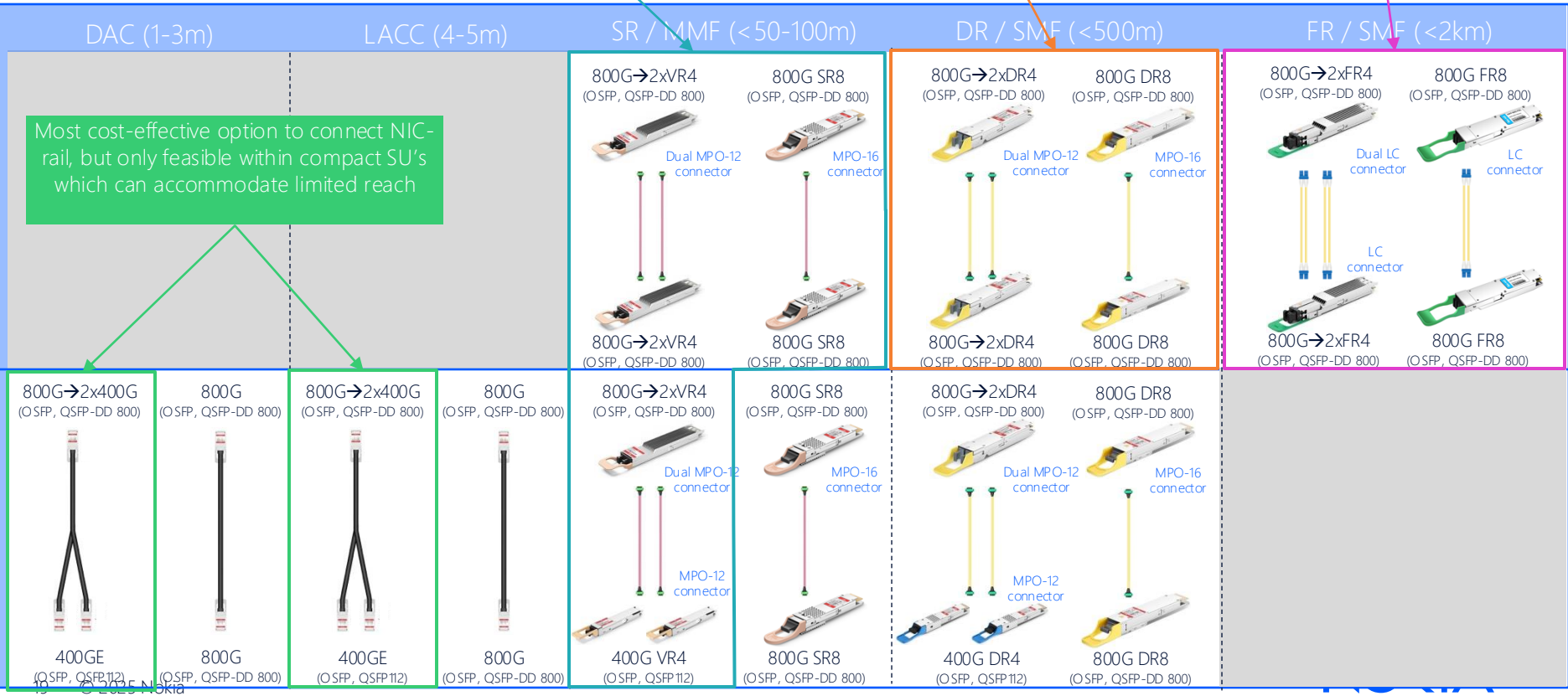
Cable Options

800GE Back-end Fabrics (400/800GE NIC attach)

If DAC/LACC reach is insufficient, then this becomes the de facto option for NIC-rail. Similarly for rail-spine and spine-core for medium sized clusters.

Used in larger clusters requiring > 50m reach to connect rail-spine and spine-core layers

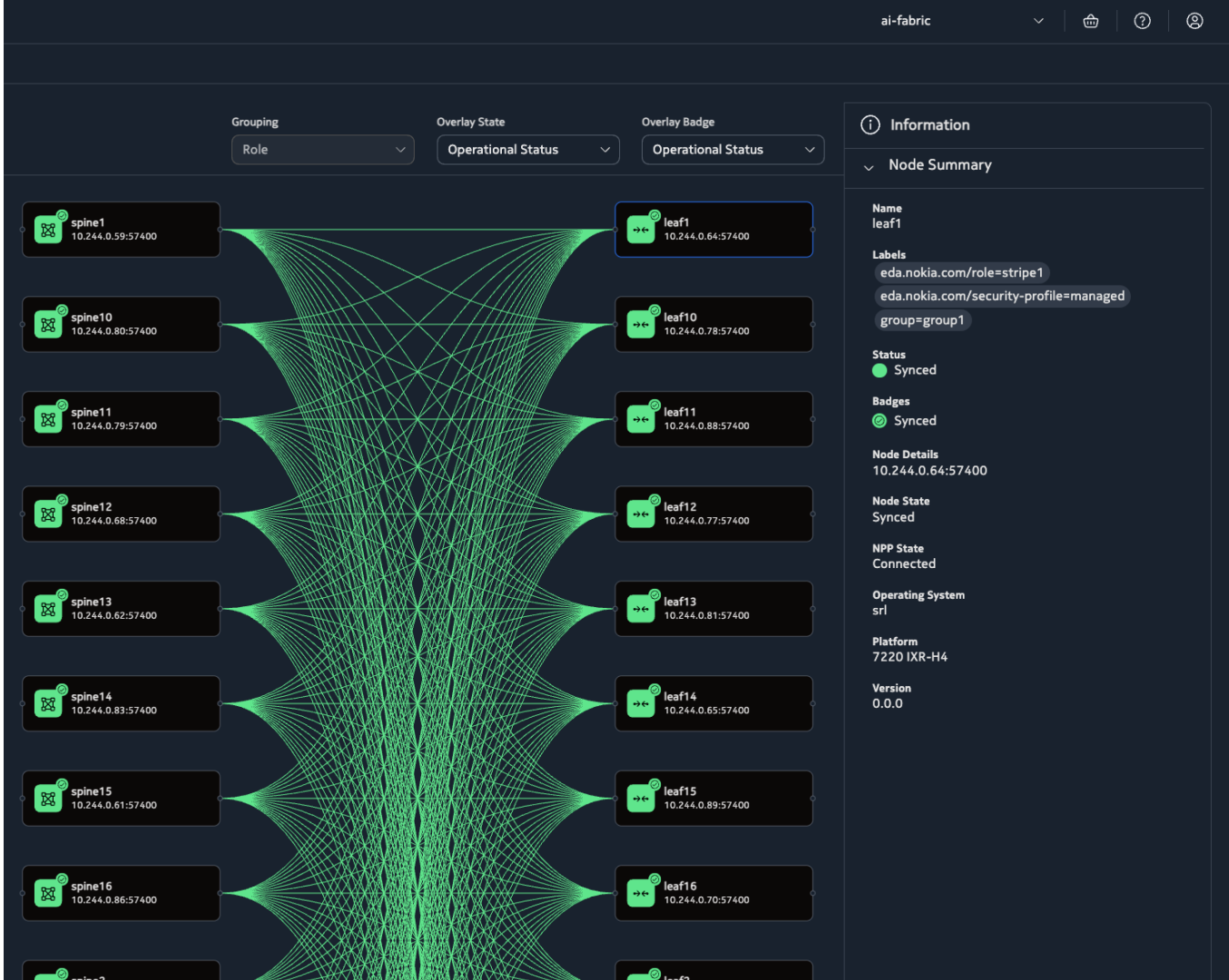
Typically, only used in multi-DC (single campus) clusters



AI Fabric Mgmt

Topology Views

- Health monitoring overlays
  - CPU
  - Memory
  - Traffic counters
  - ECN
  - PFC
  - Queue depth
- gRPC on-change subscriptions
- Multi tenancy
  - Visualize tenants and their online GPUs



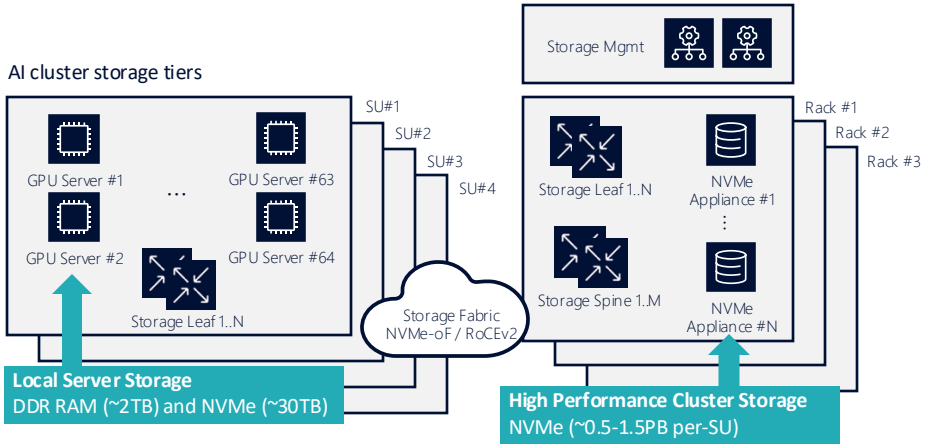
Building Modular AI Clusters

Storage & Storage Network

- High performance storage is required to support multiple functions across the AI cluster:
  - Training data ingest
  - Periodic checkpoints for resumption / fault recovery
  - Hyper-parameter search & fine-tuning
  - RAG (retrieval-augmented generation)
- Checkpointing storage "rules of thumb"
  - The larger a GPU cluster, the more likely a GPU failure → more frequent checkpointing is desired to minimize lost GPU cycles post-checkpoint

Mean Time to Failure (minutes) by cluster size (GPU count)

AI cluster training impact following GPU failure *



- High-performance NVMe based storage is deployed across the cluster to support the following key design objectives:
  - All GPU's across the cluster must be fed without waiting on I/O
  - Checkpointing writes should not consume more than 1% of total cluster training time
  - Target linear scale-out of storage capacity and throughput alongside SU build out for simplified cluster scaling

* Save early and often with multi-tier checkpointing to optimize large AI training jobs, June 2025, Google Cloud

Building Modular AI Clusters

Storage Network Design Options

Two predominant deployment models

1. Dedicated storage network
   - Aligns to Nvidia reference architecture → performance-optimized cluster design
   - Lossless Ethernet design dimensioned around storage traffic only *
   - Dedicated server NICs, storage leaf/spine switches
2. Shared in-band & storage network
   - Design adopted within many neoclouds → cost-optimized cluster design
   - Lossless Ethernet design dimensioned around storage traffic and in-band workloads *
   - Overlays required to segregate storage traffic from other front-end services

Storage performance requirements, Nvidia DGX SuperPOD Reference Architecture **

Performance Level	Work Description	Dataset Size
Good	Natural Language Processing (NLP)	Datasets generally fit within local cache
Better	Image processing with compressed images, ImageNet/ResNet-50	Many to most datasets can fit within the local node's cache
Best	Training with 1080p, 4K, or uncompressed images, offline inference, ETL	Datasets are too large to fit into cache, massive first epoch I/O requirements, workflows that only read the dataset once

Guidelines for storage performance (256 x H100 SU), Nvidia DGX SuperPOD Reference Architecture **

Performance Characteristic	Good (Gbps)	Better (Gbps)	Best (Gbps)
Peak single-GPU read	4	8	40
Peak single-GPU write	2	4	20
Peak single node read (8 GPU)	32	64	320
Peak single node write (8 GPU)	16	32	160
Single SU aggregate system read	120	320	1000
Single SU aggregate system write	56	160	496
4 SU aggregate system read	480	1280	4000
4 SU aggregate system write	240	640	2000

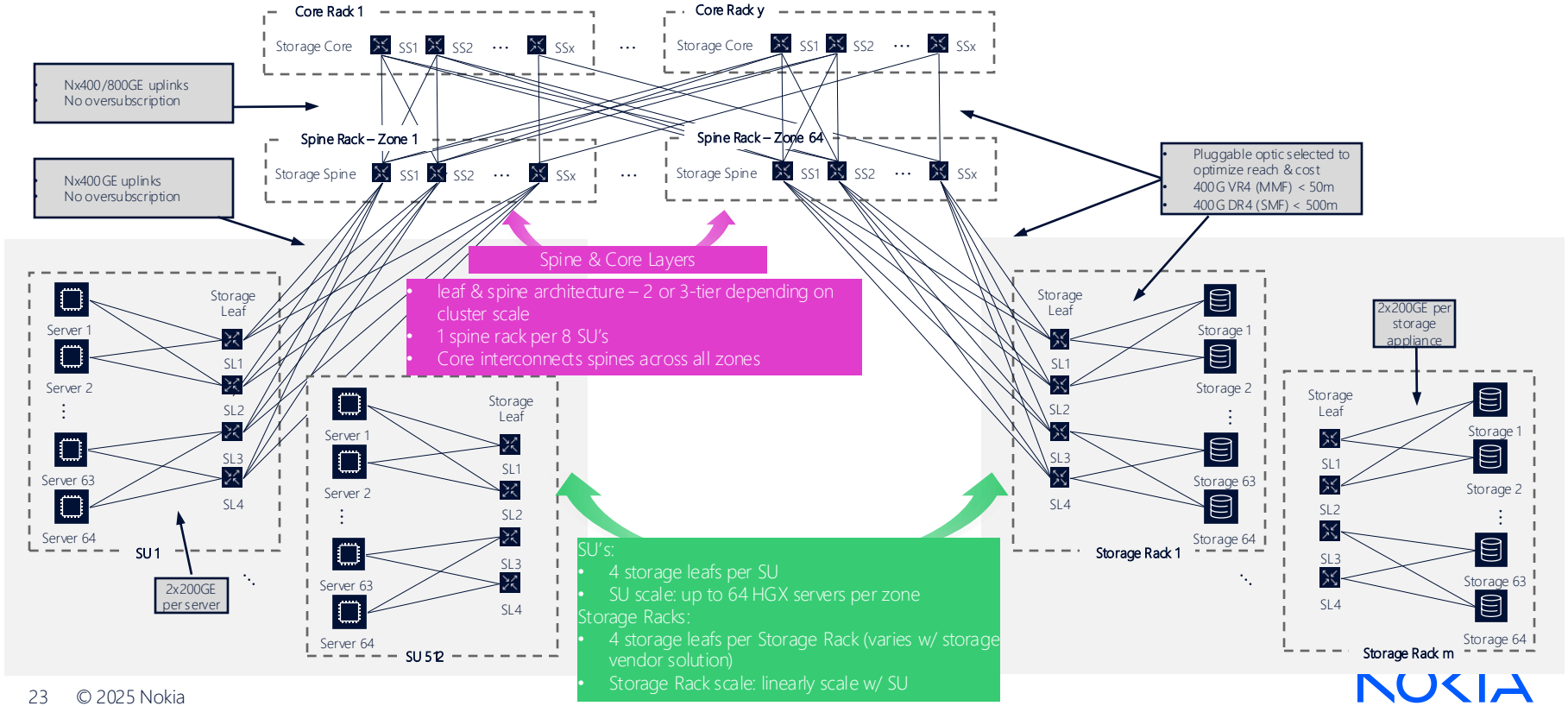
General storage network recommendations:

- Jumbo frames (9k MTU)
- ECMP hashing over IP/Ethernet fabric
- PFC/ECN enabled for Lossless Ethernet storage CoS class
- No oversubscription of dimensioned storage bandwidth
  - Include b/w headroom for front-end workloads and apply network QoS in case of shared front-end & storage network
- b/w dimensioning: know the intended AI training data – or build for the worst case

* Some storage solutions deploy a proprietary network stack across all compute which can remove the explicit requirement for a Lossless Ethernet network
** Ref: https://docs.nvidia.com/dgx-superpod/reference-architecture-scalable-infrastructure-h100/latest/storage-architecture.html

Building Modular AI Clusters

Dedicated Storage Network




... so are we there yet?

Ultra Ethernet Consortium

Ultra Ethernet Consortium

- Congestion control
- Capabilities negotiation
- Endpoint participation
- Security first class citizen
- Ethernet enhancements
- In Network Compute
- AI/HPC/storage/memory and beyond
- New DMA protocol
- Any network E2E performance
- Instrumentation and performance

… to be continued in a future session

NOKIA


NOKIA

Copyright and confidentiality

